Robotica (2002) volume 20, pp. 477–491. © 2002 Cambridge University Press
DOI: 10.1017/S0263574702004290 Printed in the United Kingdom

Dynamic bipedal walking assisted by learning

Chee-Meng Chew* and Gill A. Pratt†

(Received in Final Form: May 13, 2002)

SUMMARY

This paper presents a general control architecture for bipedal walking which is based on a divide-and-conquer approach. Based on the architecture, the sagittal-plane motion-control algorithm is formulated using a control approach known as Virtual Model Control. A reinforcment learning algorithm is used to learn the key parameter of the swing leg control task so that speed control can be achieved. The control algorithm is applied to two simulated bipedal robots. The simulation analyses demonstrate that the local speed control mechanism based on the stance ankle is effective in reducing the learning time. The algorithm is also demonstrated to be general in that it is applicable across bipedal robots that have different length and mass parameters.

KEYWORDS: Bipedal walking; Virtual model control; Reinforcement learning; Q-learning; CMAC.

1. INTRODUCTION

It is a great challenge for scientists and engineers to build a bipedal robot that can have the similar agility or mobility of a human being. The complexity of bipedal robot control is mainly due to the nonlinear dynamics, unknown environment interaction and limited torque at the stance ankle.

Many algorithms have been proposed for the bipedal walking task.[1–9] To reduce the complexity of the bipedal walking analysis, some of the researchers restricted themselves to planar motion studies, some adopted simple models or linearization approaches, etc.

Honda humanoid robots (P2 and P3)[6] are the state-of-the-art 3D bipedal walking systems. The control method is based on playing back (with modulation) trajectory recordings of human walking on different terrains.[10] Though the resulting walking execution is very impressive, such a reverse engineering approach requires iterative tunings and tedious data adaptation. These are due to the fundamental differences between the robots and their human counterparts, for example, the actuator behaviors, inertia, dimensions, etc.

MIT Leg Laboratory has recently designed and built a 3D bipedal robot called M2. This paper investigates a control approach for M2 to achieve dynamic walking behavior. The control architecture is based on a divide-and-conquer framework which is used to decompose the walking task into smaller subtasks. A control algorithm which is composed of a number of sub-algorithms can then be formulated. Reinforcement learning methods are applied to those subtasks that do not have simple solutions.

Several researchers[11–14] have proposed various learning approaches for bipedal locomotion. Benbrahim and Franklin[15] have applied reinforcement learning for a planar biped to achieve dynamic walking. They adopted a "melting pot" and modular approach in which a central controller used the experience of other peripheral controllers to learn an average control policy. The central controller and some of the peripheral controllers used simple neural networks for information storage. One disadvantage of the approach was that nominal joint trajectories were required to train the central controller before any learning began. This information is usually not easily obtainable.

In our approach, no nominal trajectories of the joints are required before the learning takes place. Since we apply learning only to smaller subtasks, the learning processes are less complex and more predictable. The learning part plays a complementary role in the overall control algorithm.

The scope of this paper covers the sagittal-plane motion-control algorithm for the biped to walk on a level ground and along a linear path. The analysis will be carried out in a dynamic simulation environment. Section 2 first describes two bipedal robots whose simulation models are used to validate the control algorithms developed in this research. Section 3 describes the proposed control architecture based on which the control algorithms for the bipedal walking task are formulated. Section 4 introduces two implementation tools that are adopted for the control algorithm. Section 5 describes the control algorithm for the sagittal-plane motion. Section 6 describes three simulation analyses which are used to illustrate different characteristics of the control algorithm.

2. THE BIPEDAL SYSTEMS

Two bipedal robots are considered in this paper. One of them is a 7-link planar bipedal robot called Spring Flamingo (see Figure 1). It is constrained to move in the sagittal plane. The total weight is about 12 kg. The legs are much lighter than the body. Each leg has three actuated rotary joints. The joint axes are perpendicular to the sagittal plane. The biped has a rotary angular position sensor at each DOF. It is used to measure the relative angular position between consecutive links. There is also a sensor to detect the pitch angle of the body. Each foot has two contact sensors at the bottom, one at the heel and one at the toe, to determine ground contact events.

The other robot is a 3D biped called M2 (Figure 2). The total weight is about 25 kg. It has more massive legs compared with Spring Flamingo. Each leg has six active

* Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260.
† F.W. Olin College of Engineering, 1735 Great Plain Ave., Needham, MA 02492–1245 (USA).

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

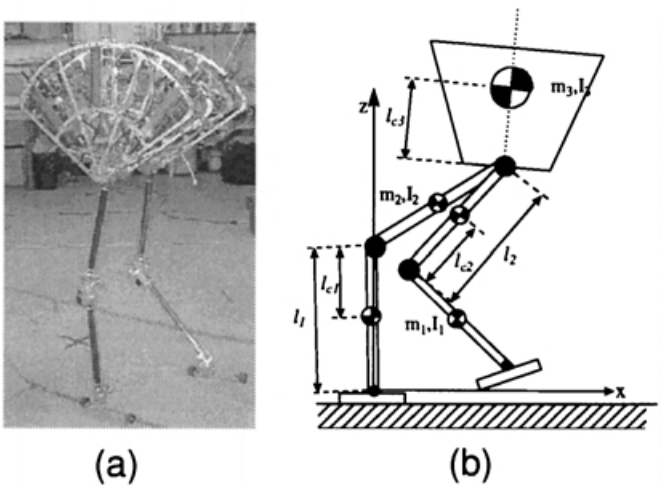


Fig. 1. 7-link 6 DOF planar biped – Spring Flamingo.

DOF of which three DOF is available at the hip (yaw, roll, pitch), one at the knee (pitch) and two at the ankle joint (pitch, roll). Each DOF has an angular position sensor to measure the relative angle between two adjacent links. Each of the feet has four force sensors (two at the toe and two at the heel) to provide the contact forces between the feet and the ground. Gyroscopic attitude sensor is mounted in the body to detect its roll, pitch and yaw angles.

In both systems, the joints are driven by force control actuators called Series Elastic Actuators.[16] They are assumed to be force sources in the simulation analysis. The model specifications for Spring Flamingo and M2 are given in Appendix A. A dynamic simulation program called SD-FAST,[17] which is based on the Newtonian mechanics for interconnected rigid bodies, is used for the dynamics simulation of these bipeds.

3. CONTROL ARCHITECTURE

A 3D bipedal walking system can be very complex to analyze from the perspectives of kinematics and dynamics. One way to overcome this problem is to partition the 3D motion space into a number of 2D motion spaces. One common approach is to project the walking motion onto the three orthogonal plane: sagittal, frontal and transverse. If we could assume that these projected motions can be independently controlled, we could decompose the overall walking control task into sagittal, frontal and transverse plane motion controls. This reduces significantly the complexity of the problem.

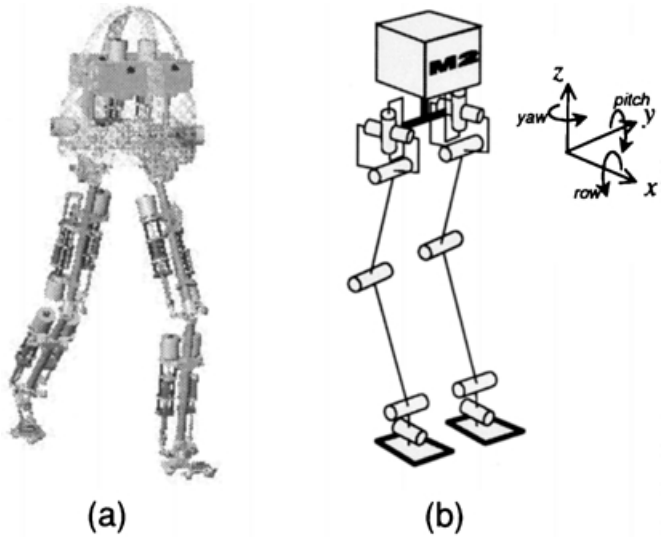


Fig. 2. Three dimensional biped M2 (CAD drawing by Daniel Paluska).

In this paper, we propose a general control architecture for 3D dynamic bipedal walking which is based on a divide-and-conquer framework. We first decompose the overall walking control task into the motion controls in each of the orthogonal planes. To further reduce the complexity, we subdivide the motion control task in each plane into small subtasks which can be assigned to the stance leg or the swing leg accordingly. Learning tools are applied to these subtasks only when they are needed.

This paper demonstrates how the control architecture can be applied to the sagittal plane motion-control task. The sagittal plane motion-control task can be decomposed into the following subtasks: (1) to maintain body pitch; (2) to maintain body height; and (3) to maintain desired walking (average) speed. It is relatively easy to achieve the first and second subtasks in the sagittal plane. However, the third subtask is much more complex. It is directly associated with gait stabilization.

There are a number of speed control mechanisms in the sagittal plane. They can be partitioned into two main classes, viz, global and local. A global speed control mechanism can achieve speed control all by itself. However, a local speed control mechanism can only achieve limited speed control. Therefore, it cannot achieve the walking speed control all by itself.

Examples of local speed control mechanisms are those that are based on the double support phase,[18] the biped's torso (using torso as a limited reaction wheel), the stance ankle, etc. In constrast, there is only one global speed control mechanism. It is based on swing leg control. If the swing leg control is badly executed, other (local) speed control mechanism may not be capable of preventing the biped from falling. Swing leg control, therefore, is the key mechanism for walking speed control. The question is how to execute the swing leg in the sagittal plane such that stable walking can be achieved.

Although the swing leg motion is an important determinant for the stabilization of the walking speed, it is difficult to obtain a closed form expression for it. This is because the dynamic equations of a biped are complex and nonlinear. This is further complicated by the unpowered DOF between the stance foot and the ground during the single support phase,[19] and the discontinuities caused by the impact at the support exchange. The overall motion of the system is usually strongly coupled to swing leg motion unless the leg's inertia is negligible.

Some research assumes that the effect of the swing leg is negligible. Thus, the system can be represented by an inverted pendulum. However, this assumption becomes invalid if the mass of the leg is significant, for example, during fast walking. The swing leg dynamics may also be utilized for walking control, for example, to create reaction torque at the hip. If its effect is ignored, no such utilization

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

is possible. The configuration of the swing leg also affects the projected center of gravity of the biped, thus changing the gravitational moment about the supporting leg. In view of the difficulty, we adopt a learning tool to assist in the swing leg motion planning.

4. IMPLEMENTATION TOOLS

This section describes the tools that are adopted for the control algorithm implementations. A control "language" for legged locomotion called Virtual Model Control (VMC)[18] is used for the motion control. This section also describes a reinforcement learning algorithm called Q-learning.[20]

4.1. Virtual model control

Virtual Model Control (VMC)[18] is a control language for legged locomotion. It allows us to work in a more intuitive space (e.g., Cartesian space) instead of a space (e.g. joint space or actuator space) that suits the robot. In the Virtual Model Control (VMC) approach, a control algorithm is constructed by applying virtual components (which usually have corresponding physical parts, for example, spring, damper, etc) at some strategic locations of the biped. The virtual components interact with the biped and generate a set of virtual forces based on their constitutive equations. Then these forces (represented by a vector $\vec{f}$) can be transformed into the joint space (represented by a vector $\vec{\tau}$) by a Jacobian matrix ${}^{A}_{B}J$:

$$\vec{\tau} = {}^{A}_{B}J^{T}\,{}^{A}_{B}\vec{f} \tag{1}$$

where ${}^{A}_{B}J$ is the Jacobian matrix which transforms the differential variation in the joint space into the differential variation of reference frame B with respect to reference frame A and the superscript^T denotes the transpose of a matrix. In the Virtual Model Control approach, frame B and frame A correspond to the action frame and reaction frame, respectively.

Equation 1 requires much lower computation compared to the equations encountered in inverse kinematics and dynamics. It is also well-behaved since, for a given force vector in the virtual component space, a corresponding torque vector can always be obtained in the joint space even if the robot is in a singular configuration.

By changing the set points of the virtual components, an inequilibrium can be created in the system. The system will then adjust itself so that a new equilibrium position is reached. The biped will behave dynamically as if the actual physical components are attached to it provided that the actuators are perfect torque sources.

4.2. Q-learning

Reinforcement learning is a class of learning problem in which a learner (agent) learns to achieve a goal through *trial-and-error* interactions with the environment. The learning agent learns only from reward information, and it is not told how to achieve the task. From failure experience, the learning agent reinforces its knowledge so that success can be attained in future trials.

Let's assume that the environment is stationary and the learning task can be modelled to be a fully observable Markov decision process (MDP) that has finite state and action sets. One popular algorithm for MDP is the Q-learning algorithm by Watkins and Dayan.[20] It recursively estimates a scalar function called optimal Q-factors ($Q^*(i, u)$) from experience obtained at every stage, where i and u denote the state and corresponding action, respectively. The experience is in the form of immediate reward sequence, $r(i_t, u_t, i_{t+1})$ $(t=0,1,2,\ldots)$. $Q^*(i, u)$ gives the expected return when the agent takes the action u in the state i and adopts an optimal policy π^* thereafter. Based on $Q^*(i, u)$, an optimal policy π^* can easily be derived by simply taking any action u that maximizes $Q^*(i, u)$ over the action set $U(i)$.

For the discounted problem, the single-step sample update equation for $Q(i, u)$ is given as follows:[20]

$$Q_{t+1}(i_t, u_t) \leftarrow Q_t(i_t, u_t) + \alpha_t(i_t, u_t)[r(i_t, u_t, i_{t+1}) + \gamma \max_{u \in U(i_{t+1})} Q_t(i_{t+1}, u) - Q_t(i_t, u_t)] \tag{2}$$

where the subscript indicates the stage number; $r(i_t, u_t, i_{t+1})$ denotes the immediate reward received due to the action u_t taken which causes the transition from state i_t to i_{t+1}; $\alpha \in [0,1)$ denotes the step-size parameter for the update; $\gamma \in [0,1)$ denotes the discount rate. Equation 2 updates $Q(i_t, u_t)$ based on the immediate reward $r(i_t, u_t, i_{t+1})$ and the maximum value of $Q(i_{t+1}, u)$ over all $u \in U(i_{t+1})$.

The convergence theorem for the Q-learning algorithm that uses lookup table representation for the Q-factors is as follows.

Theorem 4.1 (Convergence of Q-learning[20–22])
For a stationary Markov decision process with finite action and state spaces, and bounded rewards $r(i_t, u_t, i_{t+1})$. If the update equation (Equation 2) is used and $\alpha_t \in [0,1)$ satisfies the following criteria: $\sum_{t=1}^{\infty} \alpha_t(i, u) = \infty$ and $\sum_{t=1}^{\infty} [\alpha_t(i, u)]^2 < \infty$, $\forall(i, u)$; then $Q_t(i, u) \rightarrow Q^*(i, u)$ as $t \rightarrow \infty$ with probability 1, $\forall(i, u)$.

In most reinforcement learning implementation, there is an issue concerning the trade-off between "exploration" and "exploitation".[23] It is the balance between trusting that the information obtained so far is sufficient to make the right decision (exploitation) and trying to get more information so that better decisions can be made (exploration). For example, when a robot faces an unknown environment, it has to first explore the environment to acquire some knowledge about the environment. The experience acquired must also be used (exploited) for action selection to maximize the rewards.

One method that allows a structured exploration is called ϵ-greedy method.[23] In this method, the agent selects an action by maximizing the current set of $Q(i, u)$ over the action set with probability equal to $(1 - \epsilon)$ (where ϵ is usually a small number). Otherwise, the agent randomly (uniformly) selects an action from the action set $U(i)$. A greedy policy is one in which $\epsilon = 0$.

In a reinforcement learning problem, it is required to identify those actions that contribute to a failure (success) and properly discredit (credit) them. This is called the credit assignment or delayed reward problem. If an immediate

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

reward function that correctly evaluates the present action taken can be obtained, it is not necessary to consider any delayed reward. If the reward function is not precise or if the action has an eflfect that lasts more than one stage, it is required to correctly assign credit to it based on the subsequent rewards received.

In the Q-learning algorithm, the discount factor γ determines the credit assignment structure. When γ is zero, only immediate reward (myopic) is considered. That is, any rewards generated by subsequent states will not have any influence on the return computed at the present state. Usually, if a reinforcement learning problem can be posed such that γ=0 works fine, it is much simpler and can be solved more quickly. When γ is large (close to one), the future rewards have significant contribution to the return computation for the present action. The agent is said to be farsighted since it looks far ahead of time when evaluating a state-action pair.

4.3. Q-learning algorithm using function approximator for Q-factors

This subsection describes the Q-learning algorithm that uses a function approximator to store the Q-factors. The function approximator (e.g., multi-layered perceptron, radial basis functions, CMAC etc.) is used as a more compact but approximate representation of $Q(i, u)$. The purposes of using a function approximator are mainly to reduce memory requirement and to enable generalization.[23,24] These are desirable especially for high-dimensional and continuous state-action space.

In this paper, Cerebellar Model Articulation Controller (CMAC)[25] is used as a multivariable function approximator for the Q-factors in the Q-learning algorithms. CMAC has the advantage of having not only local generalization, but also being low in computation. In recent years, it has been adopted as a type of neural networks for supervised learning.[26] The Q-learning algorithm using CMAC as the Q-factor function approximator is summarized as in Figure 3.[20,23] There are many ways to implement a CMAC network. In this paper, the original Albus's CMAC implementation is adopted and the corresponding algorithm is well described in reference 27.

5. IMPLEMENTATIONS

This section describes the control algorithm for the sagittal plane motion of the bipeds. It is divided into the following subtasks: (1) the height control of the body; (2) the pitch control of the body; and (3) the forward velocity control of the body. For M2, those joints whose axes are perpendicular to the sagittal plane are ulitilized for these subtasks.

Consider the "bird-like" and "bent-leg" walking posture and assume that the desired walking height and body pitch angle are constant. The height control and body posture control subtasks can be achieved using the Virtual Model Control without the need of learning. For the horizontal speed control, it is mainly determined by the swing leg control. Since there is no analytical solution for the desired swing leg behavior, there is a need for learning, and the Q-learning algorithm is chosen to learn the key parameter(s) for the swing leg control.

The following subsection describes the implementation of the control algorithm for the stance leg and the swing leg using the Virtual Model Control approach. The subsequent subsection describes the application of the Q-learning algorithm to learn the key swing leg parameters for the sagittal plane motion so that stable walking cycle can be achieved.

5.1. Virtual model control implementation

In the sagittal plane motion control, the stance leg's joints are used to achieve the height control and body pitch control of the biped during walking. The swing leg's joints are used to execute the swing leg strategy whose parameters are obtained by learning. The control algorithm for the stance leg and the swing leg can be considered separately as follows:

The stance leg algorithm is similar to the single-support algorithm in reference 18. The height control is achieved by a virtual spring-damper attached vertically between the hip and the ground. It generates a virtual vertical force f_z at the hip. For the body pitch control, a virtual rotational spring-damper can be applied at the hip. This generates a virtual torque m_a, about the hip. The parameters of these virtual components are summarized in Table I. They are tuned by trial-and-error. The virtual forces (f_z and m_a) are transformed into the desired torques for the stance-knee joint and the

```
INITIALIZE weights of CMAC
REPEAT (for each trial):
        Initialize i
        REPEAT (for each step in the trial)
                Select action u under state i using policy (say ε-greedy) based on Q̂
                Take action u,
                Detect new state i' and reward r
                IF i' not a failure state
                    δ ← r + γ max_u' Q̂(i', u') − Q̂(i, u)
                    Update weights of CMAC based on δ
                    i ← i';
                ELSE (r = r_f)
                    δ ← r_f − Q̂(i, u)
                    Update weights of CMAC based on δ
        UNTIL failure encountered or target achieved
UNTIL target achieved or number of trials exceed a preset limit
```

Fig. 3. Q-learning algorithm using CMAC to represent Q-factors.

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

Table I. Parameters of the virtual components for the stance leg (in the sagittal plane).

Parameter	Notation
1. Spring stiffness (in z direction)	k_z
2. Damping coefficient (in z direction)	b_z
3. Spring stiffness (in α direction)	k_α
4. Damping coefficient (in α direction)	b_α

stance-hip pitch-joint (τ_k and τ_{hp}, respectively) using a transformation matrix.[18]

There are many ways to implement the swing leg strategy. This paper adopts one strategy that involves specifying a sequence of swing foot trajectory in the Cartesian space as follows:

(i) Lift the swing foot to a specific lift height, l_{lh}, from 0 to t_1 seconds. A third degree polynomial is used for this trajectory planning.
(ii) Swing the leg forward to a specific horizontal position once the foot is in the air. Again, a third degree polynomial is used for this trajectory planning.
(iii) After T_{swing} seconds, the swing foot starts to descend to the ground while maintaining its desired horizontal position.

Note that this is just one of the strategies that could be adopted for the swing leg. For example, a fifth degree polynomial may be chosen in order to specify no jerk condition.[28] Different strategies can also be adopted for the lifting and descending of the swing foot.

Given the sequence of the swing foot trajectories, two sets of linear spring-damper virtual components which are along the *X*-axis (horizontal) and the *Z*-axis (vertical) of the inertia frame respectively can be adopted. Each of them is attached between the ankle of the swing leg and the trajectory path. The virtual forces in these virtual components are then transformed by a Jacobian equation[18] to the respective joint torques of the swing leg.

The swing leg strategy adopted has several parameters that need to be set. They are lift height, swing time and end position of the swing foot. The gait stability of the dynamic walking depends on the set of parameter values chosen. In the next subsection, a reinforcement learning algorithm will be used to determine the values of selected parameters so that a walking goal can be achieved.

5.2. Reinforcement learning to learn key swing leg parameter

The key parameters for the swing leg strategy introduced in Subsection 5.1 are the lift height, swing time and the end position of the swing foot. In this paper, the lift height is set to be constant throughout the walking to simplify the problem. That is, the only key parameters considered are the swing time and the end position of the swing foot. There are three possible learning implementations. In the first implementation, the swing time is fixed and the learning agent learns the end position of the swing foot. In the second implementation, the end position of the swing leg is fixed and the learning agent learns the swing time. In the third implementation, a learning agent learns the right combinations of these parameters.

In this paper, the first two implementations are considered. The choice between the first and the second implementations depends on the terrain type. For example, if the actual foot placement or step length is not important (say in the case of the level ground walking), it is possible to simply set the swing time and learn the end position of the swing foot with reference to the body. However, if the actual foot placement is important (say in the case of the walking on stairs), it is better to specify the desired step length and learn the swing time that results in stable gait.

This subsection presents the approach whereby the decision for the swing leg parameter can be posed as a discrete-time delayed reward (or punishment) problem. By trial-and-error, the biped is supposed to learn the right swing leg parameter at each step so that long term walking is possible.

First, We need to identify the state variables for the learning task. A reward function is also set up for the learning. After that, an appropriate reinforcement learning algorithm has to be chosen. The following subsubsections are devoted to these tasks.

5.2.1. State variables. The hip height and the body pitch are assumed to be constant during the walking and hence they can be excluded from the state variables set in the learning implementation. For constant desired walking speed (average), the following variables are identified to be the state variables for the learning implementation (in the sagittal plane):

(i) Velocity of the hip in the x-direction, $\dot{x}^+$;
(ii) x-coordinate of the previous swing ankle measured with reference to the hip, x_{ha}^+;
(iii) Step length, l_{step}^+.

Superscript + indicates that the state variable is measured or computed at the beginning of a new single support phase.

The choice of the first two state variables are obvious. They represent the generalized coordinate for the body, assuming the height and the body pitch are constant. The third state variable (the step length, l_{step}^+) is required to take the dynamics of the swing leg into account. If the swing leg dynamics were negligible, this state variable will be ignorable.

5.2.2. Reward function and reinforcement learning algorithm. The biped aims to select a good value for the swing leg parameter for each consecutive step so that it achieves stable walking. A reward function that correctly defines this objective is critical in the reinforcement learning algorithm. If it is not correctly defined, the learning task may not converge. In formulating a reward function, there is usually a continuum of solutions. One also needs to decide how precise it should be.

Let's assume that there are minimum heuristics available for the implementation. A simple reward function can be formulated by defining a failure state to be one whose walking speed is above an upper bound V_u or below a lower

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

bound V_l; or the hip height z is lower than a lower bound Z_l:

$$r=\begin{cases}0 & \text{for } V_l\le\dot{x}\le V_u \text{ and } z\ge Z_l\\ R_f & \text{otherwise (failure).}\end{cases} \quad (3)$$

where R_f is a negative constant corresponding to punishment. There is no immediate reward or punishment unless a failure state is encountered. This is a *delayed reward* problem and the agent has to learn to avoid the failure states and to avoid those states that inevitably lead to the failure states.

The Q-learning algorithm that uses CMAC network as the function approximator is adopted for the implementations. The learning target is for the biped to achieve say 100 seconds of walking without encountering the failure states. Although the action sets for this application can be continuous, they are discretized so that the "maximum" Q-factor can be conveniently searched. The discretization resolution depends on the computational capability of the microprocessor and the nature of the problem. For example, in the case of the foot placement selection (see Subsection 6.1), the discretization resolution is chosen to be 0.01 m that is fine enough so as it does not affect the walking behavior nor requires high computation.

The mechanism of the learning can be briefly summarized as follows. Before learning begins, all the Q-factors are initialized to an optimistic value, say zero (by setting all the weights of the CMAC network to zero). Each learning iteration is from an initial state to either a failure state or any state in which the learning target is achieved. At the beginning of the first learning iteration, all the actions in the discrete action set will be equally good. In the event of a tie, the first action encountered in the search algorithm that has the maximum Q-factor will be adopted. Whenever a failure state is encountered, a new learning iteration is initialized. Before that, the Q-factor for the state-action pair that leads to the failure state will be downgraded. If all the actions at a given state lead to a failure state, the learning agent will avoid a state-action pair that leads to that state. As such, the learning agent gradually identifies those state-action pairs that lead to a bad outcome.

6. SIMULATION ANALYSES AND DISCUSSION

This section presents three simulation analyses. The first analysis illustrates the positive effect of a local speed control mechanism in the sagittal plane algorithm. The learning rate in the reinforcement learning algorithm will be shown to be greatly increased when the control algorithm includes such a mechanism.

The second analysis illustrates the generality of the proposed algorithm. We apply the sagittal plane algorithm to the two different bipeds introduced in Section 2. The simulation result will show that both bipeds, even though they have different length and inertia parameters, can learn equally well to walk.

In the third analysis, we modify the reinforcement learning algorithm so that it only uses a myopic learning agent instead of a farsighted one. The myopic learning agent is crucial if the robot were to have, for example, a varying velocity profile.

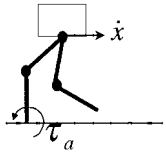


Fig. 4. A local speed control mechanism based on stance ankle joint.

6.1. Effect of local speed control on learning rate

In Section 3, the speed control mechanisms in the sagittal plane were divided into local and global. Although the local control mechanisms are not able to achieve the speed control by themselves, appropriate use of them may help to improve the performance of the control algorithm. In this subsection, the local speed control mechanism which is based on the stance ankle (Figure 4) will be demonstrated to significantly reduce the learning time.

The control law applied to the stance-ankle pitch joint to generate the required torque τ_a is as follows:

$$\tau_a=B_a(\dot{x}-\dot{x}^d). \quad (4)$$

where $\dot{x}$ is the velocity of the hip in the x-direction, B_a is a constant gain and superscript d indicates the desired value. The torque τ_a is bounded within upper and lower limits to prevent the stance leg's foot from tipping at the toe or heel. In the algorithm, a simple static analysis is used to generate the bounds:

$$\tau_{au}=K_{au}Mgl_{at-x} \quad (5)$$

$$\tau_{al}=-K_{al}Mgl_{ah-x} \quad (6)$$

where τ_{au} and τ_{al} are the upper and lower bounds of τ_a, respectively; M is the total mass of the biped; g is the gravitational constant; l_{at-x}, is the horizontal distance between the ankle and the front end of the foot; l_{ah-x} is the horizontal distance between the ankle and the rear end of the foot; K_{au} and K_{al} are positive discount factors to modulate the values of the bounds.

To test the effectiveness of the local control, a control algorithm was constructed as described in the previous section. It was applied to the simulated Spring Flamingo. The swing time T_s was fixed at 0.3 second. The Q-learning algorithm was used to learn the desired end position of the swing foot. In this particular implementation, the desired position of the swing foot was measured horizontally from the hip. It was selected from a discrete action set U $(=\{0.01n \mid for\ 0\le 0.01\ n\le 0.2 \text{ and } n\in \mathbf{Z}\})$.

The values of V_u, V_l and Z_l used in the simple reward function (Equation 3) to determine the failure conditions were mainly chosen by intuition. These parameter values (including the upper and lower bounds for the action set) were tuned by trial-and-error. The performance of the algorithm was not very sensitive to these values. For the simple reward function, a farsighted learning agent was used. The discount factor γ was set to a value close to one (say 0.9). The details of this implementation are summarized in Table II.

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

Table II. Reinforcement learning implementation for the sagittal plane: S_1.

Description	Remark
Implementation code	S_1
Swing time, T_s	constant
Desired walking speed	constant
Learning output (action)	Horizontal end position of the swing foot with reference to the hip, x_{ha}^f
Learning target	100s of walking
Key parameters:	
Reward function	Equation 3
Upper bound for the hip velocity V_u	–0.1 m/s
Lower bound for the hip velocity V_l	1.3 m/s
Lower bound for the hip height Z_l	0.5
R_f	–1
Discount factor γ	0.9 (farsighted)
Action set U	$\{0.01n \mid \text{for } 0 \le 0.01n \le 0.2 \text{ and } n \in \mathbf{Z}\}$
Policy	Greedy

The starting posture (standing posture) and initial walking speed (0.4 m/s) were the same for every iteration during the learning process. The desired walking speed was 0.7 m/s. Two learning curves are shown in Figure 5. The learning curve in dashdot-line corresponds to the case where the local control was added. It achieved 100 seconds of walking at 16th iterations. The learning curve in solid-line corresponds to the simulation result in which the local control was not added (the stance ankle was limp). From the learning curves, it can be deduced that proper application of the stance ankle torque can speed up the learning rate for the walking.

The graphs in Figure 6 show the action (x_{ha}^f) sequences generated by the learning agent (with local control at the stance ankle) for the 1st, 5th, 10th, 15th and 16th learning iterations. The corresponding forward velocity ($\dot{x}$) profiles are also included in the same figure. Except for the 16th iteration in which the learning target was achieved, all other iterations prior to the 16th iteration were terminated due to failure encounter. At the 16th iteration, the first portions (before six seconds) of the graphs for both $\dot{x}$ and x_{ha}^f were rather chaotic. After that, the motion converged to a steady cycle.

To verify that the local control at the stance ankle pitch joint did consistently result in a good learning rate, the same implementation that used the local control was applied to the simulated biped for different swing times (0.2, 0.3, 0.4 and 0.5 second). The initial walking velocity was set to 0 m/s. All other parameters were set as before. Figure 7 shows the learning curves for each of the swing times. In the worst case, the biped could achieve 100 seconds of continuous walking within 90 iterations.

In summary, the proper usage of the local speed control mechanism based on the stance ankle can help to increase the learning rate. It may be partly because the mechanism has increased the space of successful solutions. As such, the learning algorithm can find a solution much faster. The mechanism can also be viewed to possess an "entrainment"

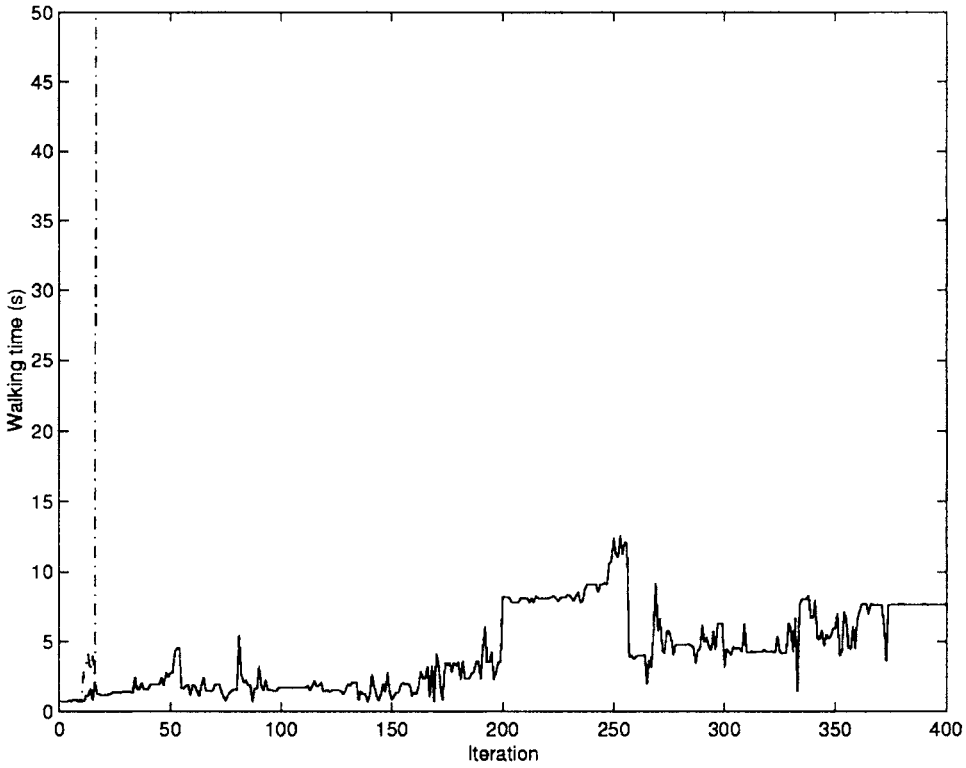


Fig. 5. The graphs show that the local control based on the stance ankle increases the learning rate. The dashdot-line graph corresponds to the learning curve with the local control at the ankle. The solid-line graph is without the local control. In both cases the starting posture (standing posture) and the initial speed (0.4 m/s) are the same for each training iteration.

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

property. This property permits the learning agent to simply generate a coarse foot placement which results in a walking speed that is sufficiently close to the desired value. Then, the mechanism try to modulate the walking speed so that the speed is even closer to the desired value. It is interesting to note that the biped was able to achieve the desired walking speed quite well even though the reward function did not take it into account. All the sagittal plane implementations in the subsequence subsection will have such a local speed control mechanism.

6.2. Generality of proposed algorithm

This subsection demonstrates the generality of the sagittal plane control algorithm in term of its applicability across bipeds of different mass and length parameters. The desired end position of the swing foot was specified based on the desired step length. Such an implementation is needed when the biped is required to select its foot placement accordingly to constraints in the terrain. One example is when the biped is required to walk on a stair.

Given a desired step length, the biped was required to learn an appropriate swing time, T_s. For a given step length, different swing times result in different postures, and hence the overall gait stability. A control algorithm very similar to that in the previous subsection was constructed for this analysis. The difference was mainly in the reinforcement learning implementation. Here, the step length was given and the learning agent learned the swing time. The learning agent was farsighted as in the previous implementation. The key parameters of the reinforcement learning implementation is summarized in Table III.

The control algorithm was applied to both the simulated Spring Flamingo (SF) and M2 (constrained to the sagittal plane). The desired step length was constant (0.3 m). The starting posture (standing position) and the initial hip velocity (0.4 m/s) were the same for each trial. The desired hip velocities were 0.7 m/s for both bipeds. The desired heights were 0.74 m for Spring Flamingo and 0.84 m for M2. Again, the local speed control based on the stance ankle was adopted to assist in the velocity control. The results are shown in Figure 8. The learning speeds were comparable, and this demonstrated the generality of the proposed algorithm in term of its application to bipeds of different inertia and length parameters. The results also demonstrated that the bipeds could achieve stable walking even though the step length was constrained.

A stick diagram of the dynamic walking of Spring Flamingo is plotted in Figure 9. The simulation data for the Spring Flamingo implementation (after the learning target has been achieved) is shown in Figure 10. Those variables with "i" preceding them were the state variables for the reinforcement learning algorithm and that with "u" preceding it was the action variable. The top graph shows the states of the state machine (state 5 and state 6 corresponded to the

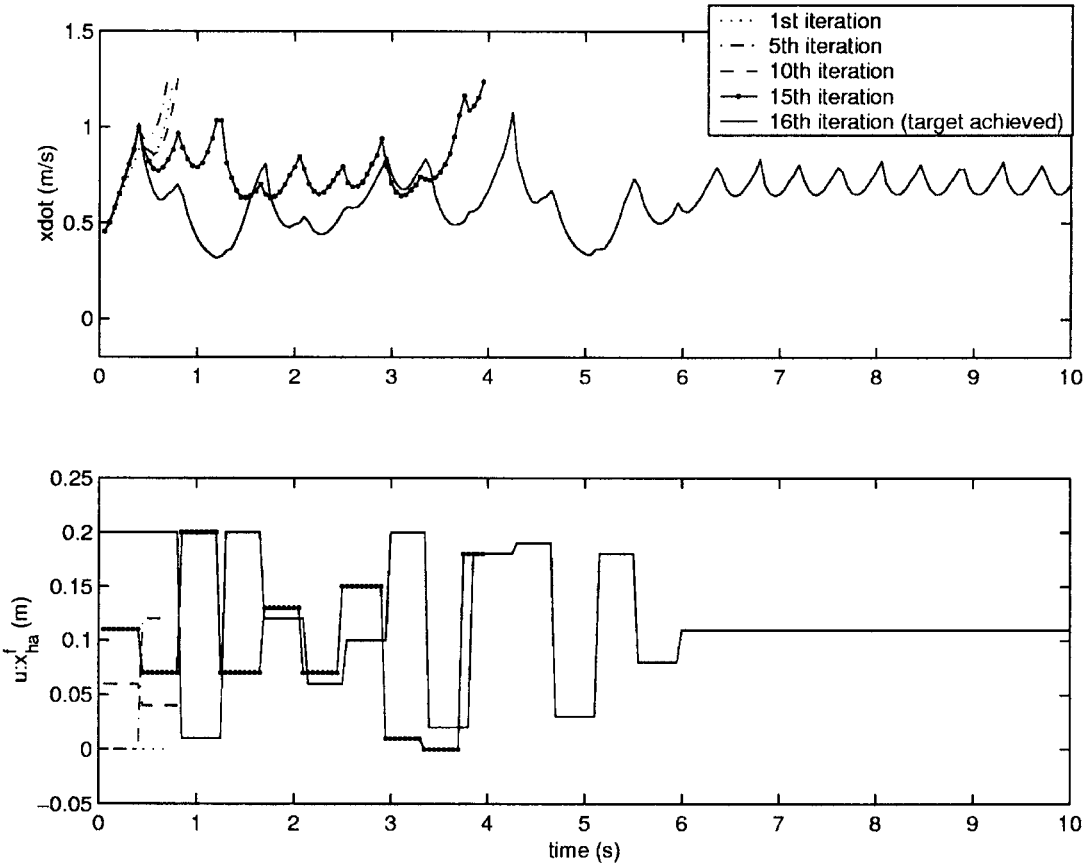


Fig. 6. The graphs show the action (x^f_{ha}) sequences generated by the learning agent (with local control at the stance ankle) for the 1st, 5th, 10th, 15th and 16th learning iterations. The corresponding forward velocity ($\dot{x}$) profiles are also included. The learning target is achieved at the 16th iteration. The starting posture (standing posture) and the initial speed (0.4 m/s) are the same for each training iteration.

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

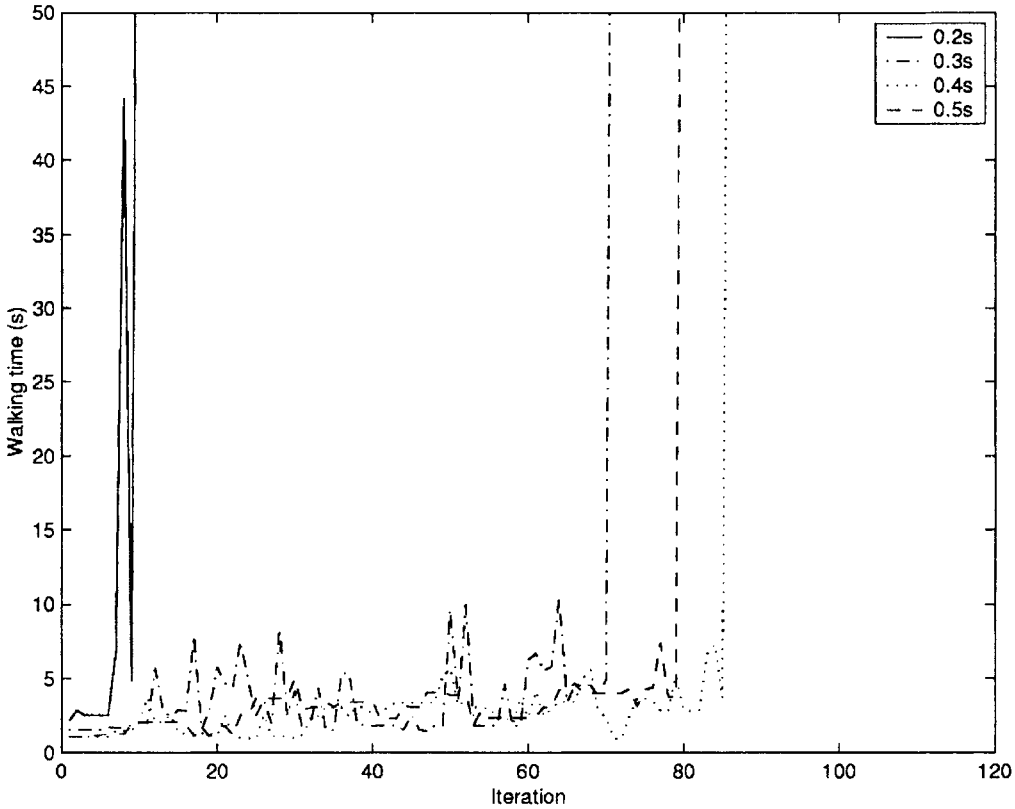


Fig. 7. The learning curves of the simulated Spring Flamingo correspond to different swing times (0.2, 0.3, 0.4 and 0.5 second). The local control based on the stance ankle is used. The starting posture (standing posture) and the initial speed (0 m/s) for each training iteration are the same for all the cases.

right support and the left support, respectively; state 7 corresponded to the right-to-left transition; and state 8 for the left-to-right transition). The horizontal dashed lines in the second and third graphs indicated the desired values.

From the second and third graphs, it is observed that the actual hip height and the forward velocity were well behaved, and they were close to the desired value. Especially for the forward velocity, although the reward function did not take into account the desired forward velocity, its average value was very close to the desired value. This outcome could be attributed to the local control at the stance ankle as discussed in Subsection 6.1.

Table III. Reinforcement learning implementation for the sagittal plane: S_2.

Description	Remark
Implementation code	S_2
Step length	constant
Desired walking speed	constant
Learning output (action)	Swing time T_s
Learning target	100s of walking
Key parameters for RL algorithm:	
Reward function	Equation 3
V_u	–0.1 m/s
V_l	1.5 m/s
Z_l	0.5 m (SF)/0.5 m (M2)
R_f	–1
Discount factor γ	0.9 (farsighted)
Action set U	$\{0.02n \mid \text{for } 0.2 \leq 0.02n \leq 1 \text{ and } n \in \mathbf{Z}\}$
Policy	Greedy

6.3. Myopic learning agent

One shortcoming of the previous implementations (using delayed reward) for the sagittal plane is that it is applicable only for constant command variables, for example, constant desired velocity, step length etc. In general, it is desirable to change walking speed and to vary the step length while the biped is walking so as to avoid certain region or obstacle (e.g. a pothole) on the ground.

To achieve these behaviors, the biped needs to evaluate its swing leg behavior based on an immediate reward rather than looking at the overall performance over several steps. In general, more heuristics or information is needed for the implementation. Sutton and Barto[23] called such a learning agent "myopic" (short-sighted) since it only tries to maximize the immediate reward.

The immediate reward is not required to be very precise. However, it must roughly represent how well the desired velocity is achieved after each swing leg control has been executed. A variable that represents the overall walking speed well is the hip velocity $\dot{x}'$ when the vertical projection of the biped's center of mass (VPCoM) passes through the stance ankle (Figure 11). VPCoM can be computed based on the kinematics information which includes the joint angles and the location of the center of mass (CoM) of each link. The mass of each link is also needed. The computation requirement is very low. Furthermore, great accuracy of the computation is not required.

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

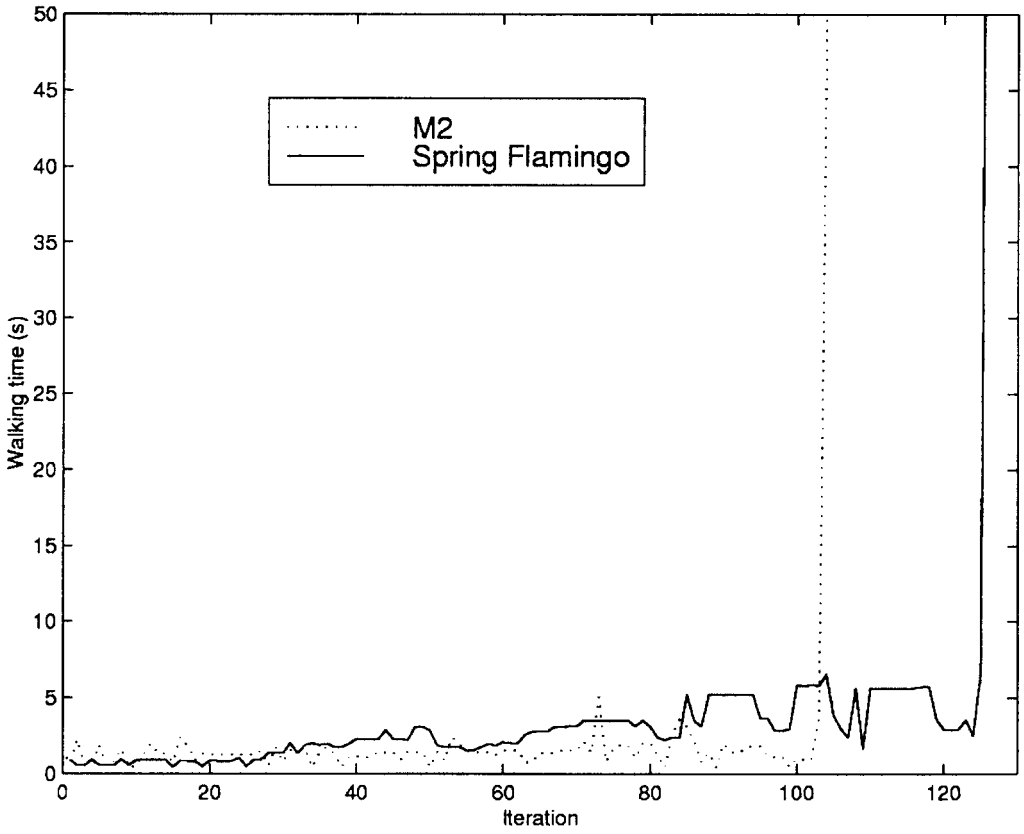


Fig. 8. Learning curves for the simulated M2 (constrained to the sagittal plane) and Spring Flamingo (using implementation S_2).

Except for those failure states, the reward function is computed based on the absolute error between $\dot{x}'$ and the desired velocity $\dot{x}_d$.

$$r=\begin{cases} -K_r \,|\, \dot{x}' - \dot{x}^d \,| & \text{for } V_l \le \dot{x} \le V_u \text{ and } z \ge Z_t \\ R_f & \text{otherwise (failure).} \end{cases} \quad (7)$$

where K_r is a positive constant.

This algorithm was implemented for the simulated M2 (constrained to the sagittal plane). The implementation details are summarized as in Table IV. The initial velocity for each iteration was set at 0.4 m/s. The desired velocity was 0.7 m/s and the desired step length was 0.25 m. The biped was required to learn the swing time, $T_s (T_s \in U = \{0.02n \mid \text{for } 0.2 \le 0.02n \le 1 \text{ and } n \in \mathbf{Z}\})$ such that it could achieve 100 seconds of walking.

Figure 12 shows a learning curve for this implementation. The biped achieved the objective (100 seconds of walking) within 20 trials. The fast learning could be attributed to the immediate reward implementation. The learning process using immediate reward was faster because the learner's action was evaluated immediately rather than delayed.

Figure 13 shows the key data during a dynamic walking of M2 after it had achieved the learning target. Those variables with "i" preceding them were the state variables for the reinforcement learning algorithm and that with "u"

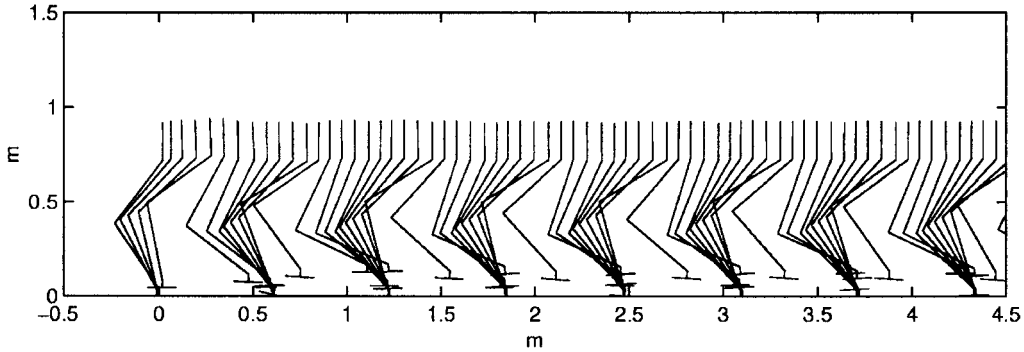


Fig. 9. Stick diagram of the dynamic walking of the simulated Spring Flamingo after the learning target was achieved (using implementation S_2). Only the left leg is shown in the diagram, and the images are 0.1 second apart.

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

preceding it was the action variable. The top graph shows the states of the state machine (state 5 and state 6 corresponded to right support and left support, respectively; state 7 corresponded to the right-to-left transition; state 8 for the left-to-right transition). The horizontal dashed lines in the second and third graphs indicated the desired values.

From the top second graph, it is observed that hip height was well-behaved (maintains a constant average value) though it fell short of the desired height (0.84 m, dashed line). The error could be corrected by either increasing the DC offset for the vertical virtual component or applying Robust Adaptive Control in place of the virtual spring-damper for the height control.[29]

From the third graph, the horizontal velocity $\dot{x}$ was also well-behaved though its average was slightly above the desired value (0.7 m/s). From the second-to-last graph, it is observed that the step length, l_{step}, was greater than the desired value (0.25 m). It was attributed to the overshooting of the swing leg control during the forward swinging phase. The precision of the swing leg trajectory tracking can be improved using Adaptive Control approaches used in manipulators.[30]

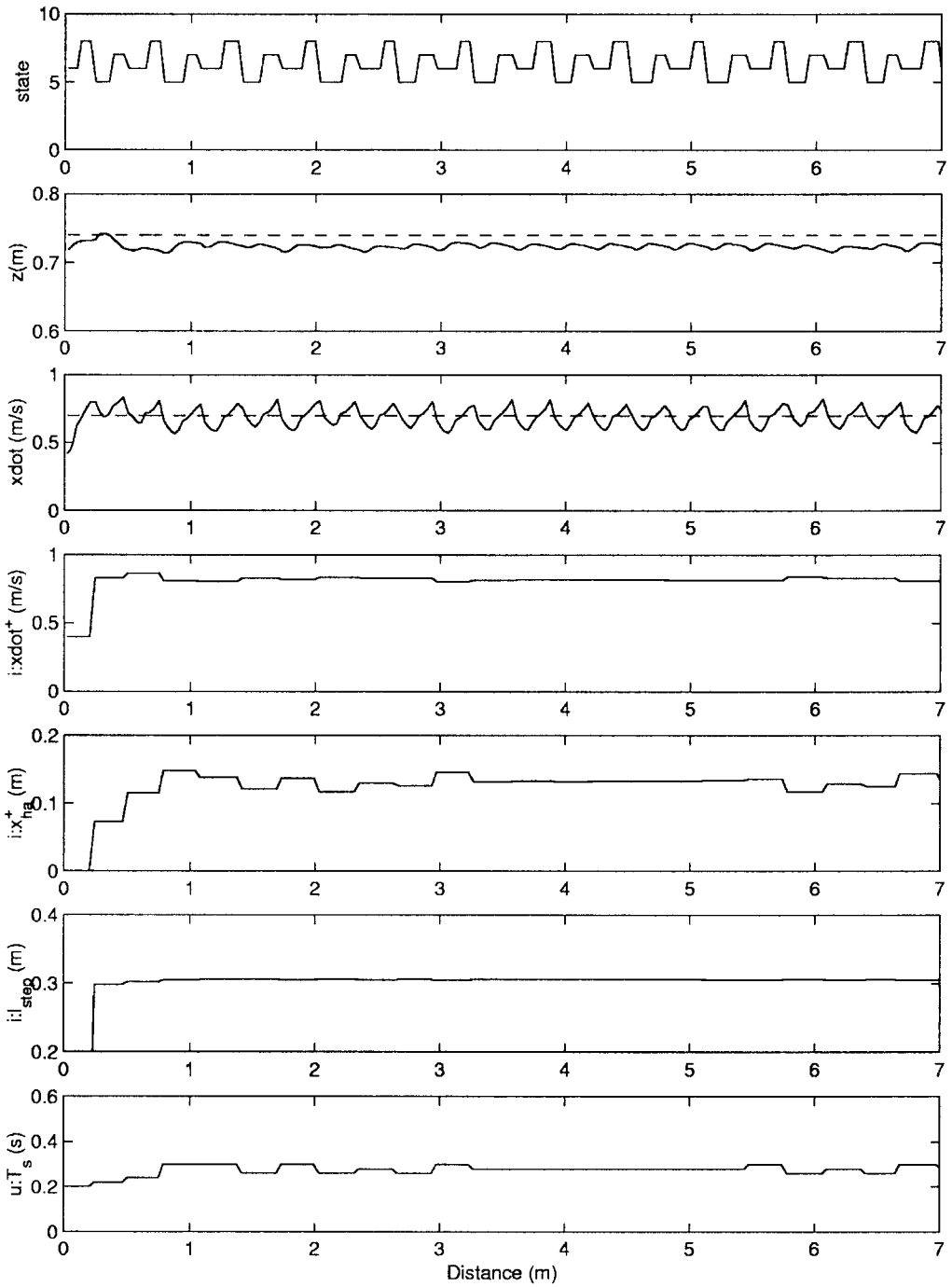


Fig. 10. The simulation data for the dynamic walking of the simulated Spring Flamingo after the learning target was achieved (using implementation S_2).

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

From the last graph, it is observed that the swing time T_s for the first step was significantly greater than the rest. It is because the initial speed of the biped is 0.4 m/s. To increase it to 0.7 m/s, the biped was required to delay the touch-down of the swing leg until it had gained sufflcient forward velocity. In summary, this implementation produces better result than the farsighted one. Though it requires slightly more computation, it allows the desired velocity and/or desired step length to be varied for each step. These behaviours are crucial especially in an unstructured environment.

7. CONCLUSIONS

This paper first presented the framework of a control architecture that was based on a divide-and-conquer approach. It illustrated how the architecture could be applied to generate the sagittal-plane motion-control algorithm. It also presented several tools used in the algorithm.

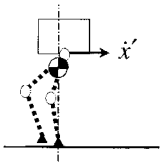


Fig. 11. $\dot{x}'$ denotes the hip velocity when the vertical projection of the biped's center of mass passes through the stance ankle. This value is used to evaluate the action taken for the swing leg.

In particular, the Q learning algorithm was applied to learn the parameter of the swing leg strategy in the sagittal plane. The Virtual Model Control approach was used to generate the desired torques for all the joints in the same plane. The resulting sagittal plane motion control algorithm was applied to the two simulated bipedal robots.

The simulation analyses illustrated the usage of the local speed control mechanism based on the stance ankle to

Table IV. Reinforcement learning implementation for the sagittal plane: S_3.

Description	Remark
Implementation code	S_3
Step length	constant
Desired walking speed	constant
Learning output (action)	Swing time T_s
Learning target	100s of walking
Key parameters for RL algorithm:	
Reward function	Equation 7
V_u	–0.1 m/s
V_l	1.3 m/s
Z_l	0.7 m
R_f	–1
K_r	0.5
Discount factor γ	0 (myopic)
Action set U	$\{0.02n \mid$ for $0.2 \le 0.02n \le 1$ and $n \in \mathbf{Z}\}$
Policy	Greedy

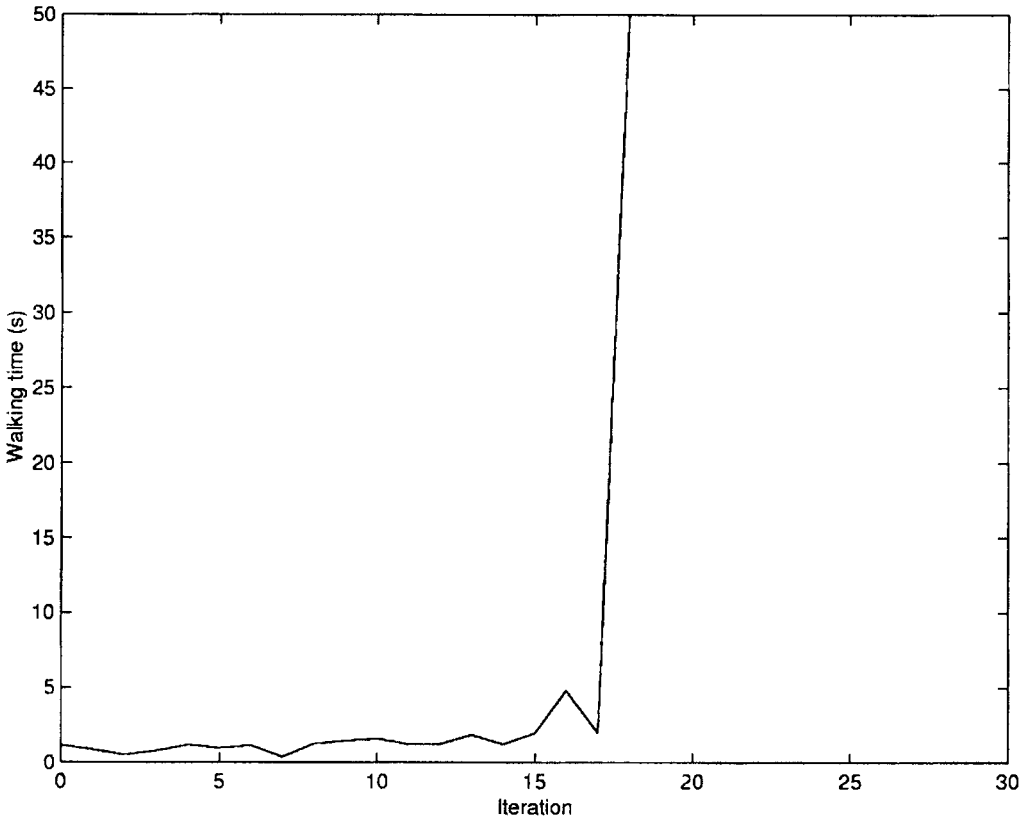


Fig. 12. A learning curve for the simulated M2 (constrained to the sagittal plane) when implementation S_3 was used.

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

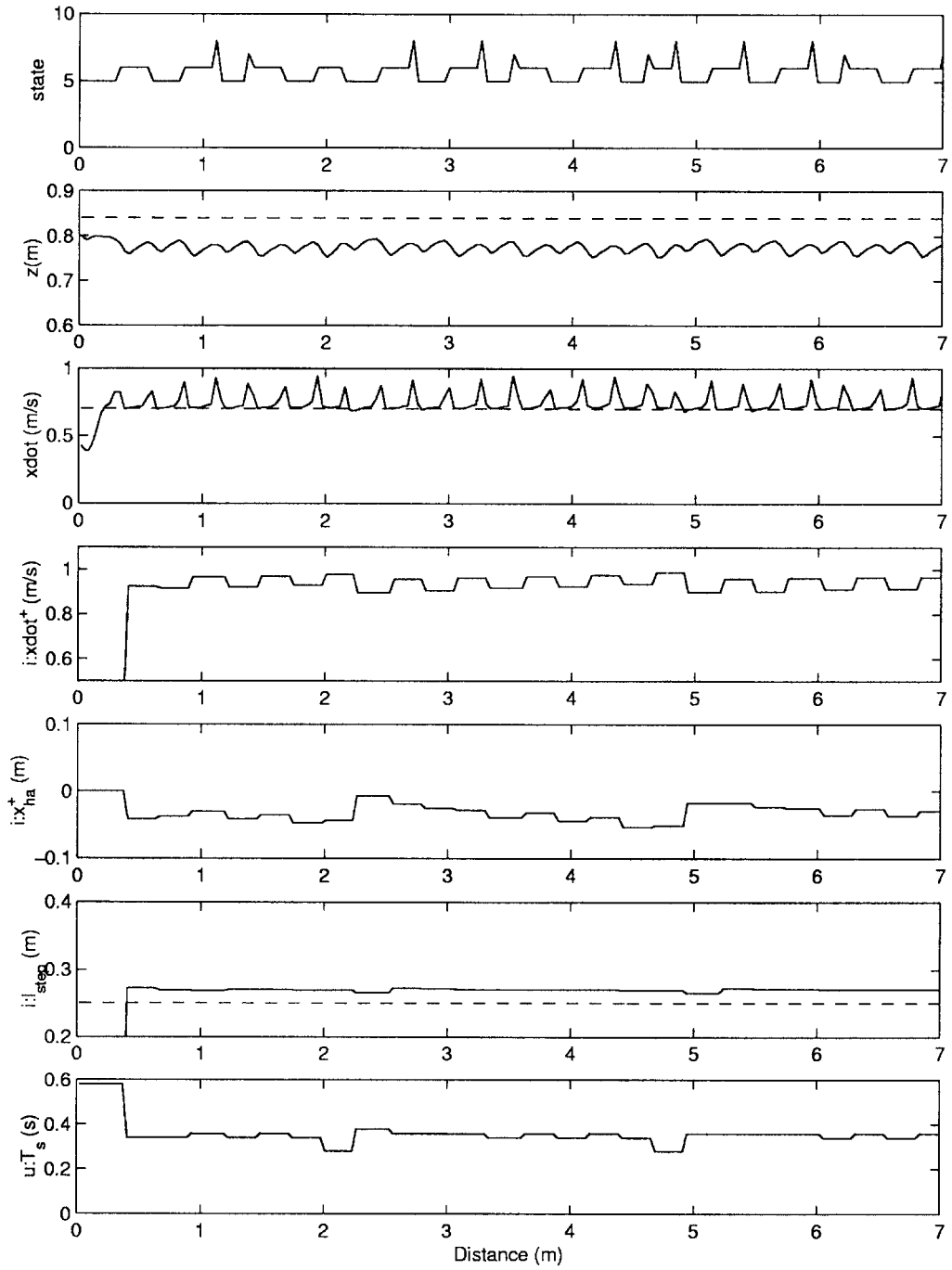


Fig. 13. The simulation data for the dynamic walking of the simulated M2 (constrained to the sagittal plane) after the learning target was achieved (using implementation S_3).

significantly improve the learning rate for the implementation. They illustrated the importance of the stance ankle even though it was not capable of achieving global speed control.

Next, the generality of the algorithm was demonstrated. The algorithm could be applied without major retunings across different bipeds that had the same structure but different inertia and length parameters. Each biped simply went through its learning process to achieve the walking motion in the sagittal plane.

We also studied the viability of adopting a myopic agent for the learning algorithm. The simulation result showed that such a learning agent was feasible. Though we only presented the simulation result for constant speed walking, the algorithm had also been shown to be applicable for variable speed walking.[31] The trade-off for this implementation was that additional information (e.g., the position of the center of mass) was needed.

In this paper, we have presented the sagittal-plane motion-control implementation. The frontal and transverse

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

plane motion-control, and the overall three-dimensional walking algorithm will be presented in subsequent paper.

References

1. F. Gubina and H. Hemami R. B. McGhee, "On the dynamic stability of biped locomotion", *IEEE Transactions on Biomedical Engineering* **BME-21**, No. 2, 102–108 (Mar., 1974).
2. S. Arimoto and F. Miyazaki, "Biped locomotion robots", *Japan Annual Review in Electronics, Computers and Telecommunications* **12**, 194–205 (1984).
3. F. Miyazaki and S. Arimoto, "A control theoretic study on dynamical biped locomotion", *ASME Journal of Dynamic Systems, Measurement, and Control* **102**, 233–239 (1980).
4. H. Miura and I. Shimoyama, "Dynamic walk of a biped", *Int. J. Robotic Research* **3**, No. 2, 60–74 (l984).
5. J. S. Bay and H. Hemami, "Modeling of a neural pattern generator with coupled nonlinear oscillators", *IEEE Transactions on Biomedical Engineering* **BME-34**, No. 4, 297–306 (April, 1987).
6. K. Hirai, M. Hirose, Y. Haikawa and T. Takenaka, "The development of honda humanoid robot", *IEEE International Conference on Robotics and Automation* (1998) pp. 1321–1326.
7. Y. Hurmuzlu, "Dynamics of bipedal gait: Part i – objective functions and the contact event of a planar five-link biped", *Journal of Applied Mechanics* **60**, 331–336 (1993).
8. Shuuji Kajita, Tomio Yamaura and Akira Kobayashi, "Dynamic walking control of a biped robot along a potential energy conserving orbit", *IEEE Transactions on Robotics and Automation* **6**, No. 1, 43l-438 (1992).
9. J. A. Golden and Y. F. Zheng, "Gait synthesis for the sd-2 biped robot to climb stairs", *Int. J. Robotics and Automation* **5** No. 4, 149–159 (1990).
10. Gill A. Pratt, "Legged robots at mit – what's new since raibert", *2nd International Conference on Climbing and Walking Robots* (1999) pp 29–33.
11. S. Kawamura, T. Kawamura, D. Fujino and S. Arimoto, "Realization of biped locomotion by motion pattern learning", *Journal of the Robotics Society of Japan* **3**, No. 3, 177–187 (1985).
12. Jin'ichi Yamaguchi, Atsuo Takanish and Ichiro Kato, "Development of a biped walking robot compensating for three-axis moment by trunk motion", *IEE International Conference on Intelligent Robots and Systems* (1993) pp 561–566.
13. H. Wang, T. T. Lee and W. A. Grover, "A neuromorphic controller for a three-link biped robot", *IEEE Transactions on Systems, Man, and Cybernetics* **22**, No. 1, 164–169 (Jan./Feb., 1992).
14. Thomas W. Miller, "Real time neural network control of a biped walking robot", *IEEE Control Systems Magazine* 41–48 (Feb., 1994).
15. H. Benbrahim and J. A. Franklin, "Biped dynamic walking using reinforcement learning", *Robotics and Autonomous Systems* **22** 283–302 (1997).
16. Gill A. Pratt and Matthew M. Williamson, "Series elastic actuators", *IEEE International Conference on Intelligent Robots and Systems* (1995) **Vol. 1**, pp. 399–406.
17. D. E. Rosenthal and M. A. Sherman, "High performance multibody simulations via symbolic equation manipulation and kane's method", *Journal of Astronautical Sciences* **34**, No. 3, 223–239 (1986).
18. J. Pratt, C.-M. Chew, A. Torres, P. Dilworth and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion", *Int. J. Robotics Research* **20**, No. 2, 129–143 (2001).
19. M. Vukobratovic, B. Borovac, D. Surla and D. Stokic, *Biped Locomotion: Dynamics, Stability, Control, and Applications* (Springer-Verlag, Berlin, 1990).
20. C. J. C. H. Watkins and P. Dayan, "Q-learning", *Machine Learning* **8**, 279–292 (1992).
21. John N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning", *Machine Learning* **16**, No. 3, 185–202 (1994).
22. T. Jaakkola, S. P. Singh and M. I. Jordan, "On the convergence of stochastic iterative dynamics programming algorithms", *Neural Computation* **6**, 1185–1201 (1994).
23. R. S. Sutton and A. G. Barto, *Reinforcement Learning An Introduction* (MIT Press, Cambridge, MA, 1998).
24. D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming* (Athena Scientific, Belmont, MA, 1996).
25. J. S. Albus, *Brain, Behavior and Robotics* (BYTE Books, McGraw-Hill, Peterborough, NH, 1981). Chapter 6, pp. 139–179.
26. W. T. Miller, F. H. Glanz and M. J. Carter, "Cmac: An associative neural network alternative to backpropagation", *IEEE Proceedings* No. 78, 1561–1567 (1990).
27. Thomas W. Miller and Filson H. Glanz, "The university of new hampshire implementation of the cerebellar model arithmetic computer – cmac", *Unpublished reference guide to a CMAC program* (August, l994).
28. T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model", *Journal of Neuroscience* **5**, No. 7, 1688–1703 (1985).
29. Chee-Meng Chew and Gill A. Pratt, "Adaptation to load variations of a planar biped: Height control using robust adaptive control", *Robotics and Autonomous Systems* **35**, No. 1, 1–22 (2001).
30. Jean-Jacques E. Slotine and Weiping Li, *Appiled Nonlinear Control* (Prentice-Hall, Cambridge, MA, 1991).
31. Chee-Meng Chew, "Dynamic Bipedal Walking Assisted by Learning", *PhD dissertation* (Massachusetts Institute of Technology, Sept 2000).

APPENDIX

A. Specifications for Spring Flamingo and M2

Table V. Specifications of Spring Flamingo.

Description	Value
Total Mass	12.04 kg
Body Mass	10.00 kg
Thigh Mass	0.46 kg
Shin Mass	0.31 kg
Foot Mass	0.25 kg
Body's Moment of Inertia about COM	0.100 kgm^2
Thigh's Moment of Inertia about COM	0.0125 kgm^2
Shin's Moment of Inertia about COM	0.00949 kgm^2
Foot's Moment of Inertia about COM	0.00134 kgm^2
Thigh Length	0.42 m
Shin Length	0.42 m
Ankle Height	0.04 m
Foot Length	0.15 m

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press

Table VI. Specifications of M2.

Description	Value
Total Mass	25.00 kg
Body Mass	12.82 kg
Thigh Mass	2.74 kg
Shin Mass	2.69 kg
Foot Mass	0.66 kg
Body's Principal Moment of Inertia	
x-axis	0.230 kgm^2
y-axis	0.230 kgm^2
z-axis	0.206 kgm^2
Thigh's Principal Moment of Inertia	
x-axis	0.0443 kgm^2
y-axis	0.0443 kgm^2
z-axis	0.00356 kgm^2
Shin's Principal Moment of Inertia	
x-axis	0.0542 kgm^2
y-axis	0.0542 kgm^2
z-axis	0.00346 kgm^2
Foot's Principal Moment of Inertia	
x-axis	0.000363 kgm^2
y-axis	0.00152 kgm^2
z-axis	0.00170 kgm^2
Hip Spacing	0.184 m
Thigh Length	0.432 m
Shin Length	0.432 m
Ankle Height (from ankle's pitch axis)	0.0764 m
Foot Length	0.20 m
Foot Width	0.089 m

https://doi.org/10.1017/S0263574702004290 Published online by Cambridge University Press